User Manual for SAOimage

A Pseudocolor Display Program for Grayscale Images

M. VanHilst*

Smithsonian Astrophysical Observatory
Harvard/Smithsonian Center for Astrophysics
Cambridge, Massachusetts

1 January 1991

*Work supported through NASA Contracts NAS8-30751, NASW-3722, NASS-29350 and NASS-30934

Contents

1 SAOimage command line arguments

The *command line* is the line you type to start up the *SAOimage* program. Following the name of the program, you can include the name of a file to read as well as various parameters and their settings. Except for the filename, all settings are identified by a *switch* beginning with a '`-`' (a hyphen). Some switches are sufficient by themselves, some switches must be followed by a fixed number of arguments (usually numerical), and some switches may be followed by arguments but do not require them. These are explained below.

There is no required order to the switches. The filename can also appear anywhere on the command line. *SAOimage* assumes that any token, which is not part of a recognizable switch, is the filename.

Many switches have two names, one literal and one abbreviated. In such cases the names are completely interchangeable. The longer names might be used in a script where a later reader might wonder what the switches do. The abbreviated names save typing on the command line.

Once *SAOimage* is running, you may enter a new command line, changing some of the settings and/or reading in a new image from disk. See the descriptions for the `new` menu under `etc` in section 2.2 and the `N` key in section 10.1. A few switches cannot be changed from their initial settings (e.g. `-display` to change the display server). These are marked by `**` below.

The following is a list of the command line switches for *SAOimage*:

`-blue **`

Set the color of all graphics to be blue. Some inexpensive systems use a monochrome monitor connected to one of the three color outputs on the computer. That color must be specified to make the graphics visible.

`-bordercolor <colorname> **`
`-bc <colorname>`

Specify the color of all subwindow borders. The color name must be a recognized X color (there are many). This is a style issue.

For instance, one can turn on green borders by typing

```
-bc green
```

`-byteswap`
`-bswap`

Switch the bytes order between big-endian and little-endian order. This may be needed where data has been copied from another machine or if there is some confusion about the FITS file format. This switch toggles the previous setting.

`+/-coord`
`+/-ct`

Set the coordinate tracking state initially on or off. In coordinate tracking, the coordinates of the mouse and value of the pixel under it are printed in the upper-left text area, above the main display window.

`-display <X server name>:<X server name extension>.<screen number> **`
`-d <X server name>:<X server name extension>.<screen number> **`

Specify the name of the X display server. This makes it possible to run the *SAOimage* program on a machine other than the one connected to your display screen, with no difference in appearance or use. By default, *SAOimage* gets the server name from the `DISPLAY` environment variable. See the *xhost* manual page for more details. The display server cannot be changed once *SAOimage* is running.

For example, to connect to the display server on `cfa241`, one would use

```
saoimage -d cfa241:0.0      or, equivalently
saoimage -d 128.103.41.241:0.0
```

`-dfits`

Image file is a FITS file (see `-fits`), but in unexpected byteswapped order. The FITS standard is not swapped, but some naive VAX applications may swap it (see `-bswap`).

`-fbconfig <filename>`

Specify an alternate frame buffer configuration file for use with IRAF. By default, the file installed with *SAOimage* (`/usr/local/lib/imtoolrc`) is used.

`-fits`

Image file is a FITS file. If the image filename ends in `.fits`, this switch is not necessary. Only `T=SIMPLE` array types are supported. The header `BITPIX` card must be 8 (unsigned byte), 16 (signed short), 32 (signed int), -32 (float), -64 (double), or -16 (unsigned short). (The last two are not recognized standards). IEEE floats are not converted if that is not the machine format.

`-gd [<width>x<height>][+-<x>+-<y>] **`

Specify the size of the image display subwindow and/or the screen position of *SAOimage*. The format is a standard X geometry statement. This switch works like `-geometry`, except that width and height (if given) are applied to the display subwindow. The overall *SAOimage* window is sized accordingly.

For example, to create an *SAOimage* in the upper right with a display window that exactly fits a 320x512 FITS image, type

```
saoimage -gd 320x512-5+0 m51.fits
```

`-geometry [<width>x<height>][+-<x>+-<y>] **`
`-g [<width>x<height>][+-<x>+-<y>] **`

Specify the size and/or the screen position of *SAOimage*. The format is a standard X geometry statement. Both size and position may be specified, or only the size or the position. Width and height refer to the dimensions of *SAOimage*'s desktop window (see `-gd` for sizing just the image display window). `+x` and `+y` refer to the upper left corner in screen coordinates. `-x` positions the right edge from the right edge of the screen. `-y` would positions the lower edge from the bottom of the screen. Width and height below a minimum size are defaulted to the minimum. Specifying the default minimum size (`-geometry 0x0`) also triggers *SAOimage* to use smaller dimensions for its internal windows. Once *SAOimage* is running, use the window manager's normal size and move mechanisms to make adjustments to *SAOimage*'s main window.

For instance, to create minimum sized *SAOimage* near the upper left corner, type

```
saoimage -geometry 0x0+10+20
```

To create a 500x700 *SAOimage* window in the lower right corner, type

```
saoimage -geometry 500x700-5-5
```

To create a default sized *SAOimage* window in the upper right corner, type

```
saoimage -geometry +0-5
```

`-green **`

Set the color of all graphics to be green. See `-blue`.

`-histeq`

See section 3.8.5 on histogram equalization.

`-horizgraph **`
`-hg`

Use a horizontal auxiliary color graph window, with a color bar along the bottom. See `-vg`.

`-i2 <width> <height>`
`-shortarray <width> <height>`

Image file is a signed short integer array file of the given dimensions. If the file is square and has no added padding, the dimensions are not necessary.

`-i4 <width> <height>`
`-longarray <width> <height>`

Image file is a signed long integer array file of the given dimensions. If the file is square and has no added padding, the dimensions are not necessary.

`-idev <pipename>`

Specify the name of the named pipe used for listening. The default is `/dev/imt1o`, which is the default used by IRAF. See `-odev`.

`+/-imtool`

Open/close the named input pipe connection and wait for input from IRAF. When open, *SAOimage* emulates IRAF's *imtool*. IRAF's image loading and cursor read-back functions are supported. Unlike *imtool*, *SAOimage* has only one frame buffer; IRAF's frame buffer numbers are ignored. Listening on the pipe is possible even while reading image files directly. The connection may be opened, closed, or re-opened at any time. When supported, the default mode is commonly to start with the IRAF connection open. See `-idev`, `-odev` and `-pros`.

`-linear`

See section 3.8.1 on linear scaling.

`-log [<exponent for exponential curve>]`

Set the scaling mode to log (exponential), and set the exponent for the curve function e^n, if given. See section 3.8.3.

`-lowerleft`
`-ll`

First pixel in file represents the lower left of the image, assuming the lines of input run left to right on the screen. This is the IRAF standard and the *SAOimage* default. See `-rotate`, `-upperleft`, and `-zero`.

`-lprbuttons`

Include the button menu in the hardcopy image (only on color workstations). The default on color workstations includes the area above the button panel, but excludes the buttons.

`-mag <magnification>`

Set the magnification factor of the magnifier. This factor relates the magnifier to the magnification of the display window. The default is 4; the magnifier magnifies the image to 4 times the magnification of the main display window (but never less than zoom 1 of the original data).

`+/-magnifier`
`+/-mt`

Set the magnifier tracking state initially on or off. With magnifier tracking, the magnifier window is continuously updated to show a magnification of the image the image under the mouse.

`-max [<maximum value for scaling>]`

Set the maximum for the image value range used to compute scaling. The default is to take the maximum from the image shown in the display window. `-max` with no value resets the default. If the maximum value in the displayed image is lower than the given maximum, the image's maximum vale is used for the scaling range.

`-min [<minimum value for scaling>]`

Set the minimum for the image value range used to compute scaling. The default is to take the minimum from the image shown in the display window. `-min` with no value resets the default. If the minimum value in the displayed image is higher than the given minimum, the image's minimum vale is used for the scaling range.

`-mtf`

Give the button panel a chiseled look popularized by HP's widget set. This appearance may contrast less with other applications being used at the same time.

`-name <filename>`

This switch is only needed if the filename starts with a number or might otherwise be recognized as a switch.

`-odev <pipename>`

Specify the name of the named pipe used for sending feedback. The default is `/dev/imt1i`, which is the default used by IRAF. See `-idev`.

`-oif`

Image file is an IRAF image header file in OIF format. If the image filename ends in `.imh`, this switch is not necessary. IRAF STF and QPOE formats are not supported. Complex data cannot be handled. The data must have at least 2 dimensions. Only the first plane of multidimensional images is read. The data file is read directly by *SAOimage* (see `-imtool` and `-pros`).

`-one`

The file coordinate of the first pixel is $(1, 1)$. The real coordinates of the center of the first pixel are $(1.0, 1.0)$. This is the IRAF standard and the default for *SAOimage*. The second pixel is $(2, 1)$. See `-zero`.

`-palette <number of colors in display colormap palette>`
`-p <number>`

Specify the number of read/write color cells to reserve. On color workstations, *SAOimage* reserves color cells in the default colormap for its own use (see section 5). *SAOimage* reserves as many color cells as it can get, up to the number given (the default is 200). If the number given is negative, *SAOimage* comes up in overlay mode, using $1/2 + 2$ of the color cells for overlays and graphics. In verbose mode (see `-verbose`), *SAOimage* tells you how many cells it is able to use for display colors. This number can be re-entered at run-time, unless `-palette 1` is given, in which case *SAOimage* stays in halftone mode.

`-panboxav`
`-panboxsum`
`-panboxsamp`
`-panboxmax`

These switches select the kind of image reduction used to fit a picture of the entire image into the pan window. Each pixel is computed from a block of image pixels by averaging, summing, sampling, or taking the maximum. The default is to show the maximum from each block. When zooming in the main display involves reduction, subsampling is always used.

`-pros`

This command is virtually identical to `+imtool`. The difference occurs when the user writes the saved regions to a disk file. *imtool* emulation includes writing only an IRAF list file giving center coordinates only. With `-pros`, *SAOimage*'s normal region descriptor file will be written in place of the simpler list file. One may switch between this mode and `+imtool`, or close the IRAF connection with `-imtool`.

-quiet
-q

Disable *verbose* mode. See -verbose.

-r4 <width> <height>
-floatarray <width> <height>

Image file is a real$*$4 array file of the given dimensions. If the file is square and has no added padding, the dimensions are not necessary.

-r8 <width> <height>
-doublearray <width> <height>

Image file is a real$*$8 array file of the given dimensions. If the file is square and has no added padding, the dimensions are not necessary.

-red **

Set the color of all graphics to be red. See -blue.

-rmax [<maximum value for reading>]

Set maximum value for reading from the image file. This value is used as the maximum value when images are pre-scaled to fit the 16 bit (signed short) working buffer. See section 3.8.

-rmin [<minimum value for reading>]

Set minimum value for reading from the image file. This value is used as the minimum value when images are pre-scaled to fit the 16 bit (signed short) working buffer. See section 3.8.

-rotate <1,2,or 3>
-rot <1,2,or 3>

Rotate the image 90, 180, or 270 degrees (respectively) before displaying it. Rotation is applied after conversion to a lower left coordinate system (-ll) if such conversion is also requested. This is useful for images when the CCD was not mounted North-up.

-scalebias <scale> <bias>
-sb <scale> <bias>

The data in the image file should be scaled and biased to get the true image value (TrueValue = (scale $*$ FileValue) + bias). This cannot be used with the -fits image type (scale and bias are in the FITS header), nor with -imtool or -pros (they are passed by IRAF).

-skip <bytes>
-sk <bytes>
-header <bytes>

Skip over the given number of bytes at the head of the file before reading data. This is used to skip header information or the first image if two images are stored in one file.

-sqrt [<inverse of exponent for geometric curve>]

Set the scaling function to square root (geometric), and set the inverse of the exponent for the $x^{1/n}$ curve, if given. See section 3.8.4.

`-u1 <width> <height>`
`-chararray <width> <height>`

Image file is an unsigned byte array file of the given dimensions. If the file is square and has no added padding, the dimensions are not necessary.

`-u2 <width> <height>`
`-ushortarray <width> <height>`

Image file is a unsigned short integer array file of the given dimensions. If the file is square and has no added padding, the dimensions are not necessary.

`-upperleft`
`-ul`

First pixel in file represents the upper left of the image, assuming the lines of input run left to right on the screen (see `-rotate` and `-lowerleft`). This switch does not override IRAF WCS image coordinates.

`+/-verbose`
`+/-v`

Set *verbose* mode on or off. In verbose mode, informative statements are printed to the terminal window when various actions are taken. The default mode is to be verbose.

`-vertgraph **`
`-vg`

Use a vertical auxiliary color graph window, with a color bar along the left side. See -hg.

`-wrap [<number of wraps within scaling range>]`

Set the scaling mode to wrapped linear, and set the number of wraps for this mode, if given. See section 3.8.2.

`-zero`

The file coordinate of the first pixel is $(0,0)$. The real coordinates of the center of the first pixel are $(0.5,0.5)$ which makes the very edge $(0,0)$. This is the standard coordinate system for image displays, but not the default for *SAOimage*. The second pixel is indexed $(1,0)$. See -one.

2 SAOimage button menus and submenus

SAOimage uses button panels for selecting the most common actions and modes. The button panel is organized in two rows. The upper row of buttons selects the main mode. The lower row of buttons controls modes, actions, and selections specific to the current main mode.

The upper row of buttons, referred to as the *main menu*, or just the *menu*, is always visible. The lower row of buttons, referred to as the *submenu*, changes to correspond to the mode selected in the main menu. For some main menu modes (Color and Cursor), there are two interchangeable submenus, toggled by the button on the end (cmap and region, respectively).

Main menu modes all correspond to modes of interaction for the mouse and its buttons. For example, when in Color mode, dragging the mouse in the display window manipulates the color map, while in Cursor mode, the same action affects the drawing of a cursor. Making a selection in the main menu only changes modes and submenus. Selections in submenus often do cause specific actions to be taken. Most of these actions are easily reversible. The user is especially warned that in the etc submenu, the QUIT button does what it says, and the print button sends a large PostScript file to the user's printer.

2.1 Where to find more information

For additional information about the Scale menu, see section 3. The mouse interactions that are possible in the Scale menu are described in section 4 dealing with blinking.

The Color menu is described in section 5. Note that on color workstations, the Color menu has two different submenus, interchangable by using the cmap button. One can also switch to non-color mode by toggling the mono button.

Details about the Cursor menu are covered in sections 6 and 7. The Cursor menu has two submenus, interchangeable by pressing the region button.

For the Pan menu, see section 8.

2.2 Explanation of items in the "etc" submenu

The etc menu groups a numbers of different functions.

new brings up a command line editor. A new command line can be entered to read a new image and also to change certain internal parameters. The previous command line is initially presented, should you just want to edit it. Type `Ctrl-N` if you want to start with an empty line. `Ctrl-P` recalls previous lines. Type `Ctrl-C` to exit the command line editor without taking any new actions. In most modes, typing `N` has the same effect as the new button. See sections 1 and 10 for further details.

track enables the tracking mode. In tracking mode, the magnifier window magnifies the image under the mouse when it is in the main display window. Tracking mode also controls whether the the color graph gets updated continuously when in Color mode, or only at the completion of some action. Whatever mode tracking is in, the opposite can be temporarily invoked by holding down the `Shift` key, or using the `CapsLock` key. The default is to be initially on (see section 1 for how to change this).

coord enables coordinate tracking mode. In coordinate tracking mode, the coordinates and pixel value of the pixel under the mouse are printed, and continuously updated, when the mouse is over the main display window. This mode is also temporarily negated by using the `Shift` key. The default is to be initially on (see section 1 for how to change this).

verb controls verbose mode. When in verbose mode, *SAOimage* prints information about the cursors or image to the terminal window of the parent process from which *SAOimage* was invoked. This may or may not be desired. The default is to be initially on (see `-verbose` in section 1 for how to change this).

print creates (and prints) a PostScript hardcopy. See section 11 for details.

raise tells the window manager to raise *SAOimage*'s windows. Most window managers in use now have their own features to accomplish this. (Typing the `R` key also has this effect in most modes).

QUIT terminates *SAOimage*.

2.3 User interface issues

The user-interface of *SAOimage* has been the subject of many questions and comments. A common question is: "Why isn't it *Motif*?" *SAOimage*'s button menu was developed under X10 and predates any X toolkits. While successors to *SAOimage* will undoubtably use toolkits like *Motif* or *OpenLook*, there is still justification for an xlib-only design.

1. *SAOimage* is distributed completely free of any license restrictions. It supports a community characterized by limited funds and a wide variety of machines. Using *SAOimage*, and making one's own modifications, requires only that one have X11.

2. *SAOimage* runs well on machines with limited memory and/or processing power. These are still issues for older workstations and PC's and Mac's running UNIX.

3 Scaling image file values to the range of the display color map

Workstation displays pose a common problem for imaging applications: how to display an image with a large range of values on a screen with a small range of colors. Since there aren't enough colors to use one for each image value, groups of different image values must all be assigned the same color. The process of grouping image values into a smaller set of values is called *scaling* or *binning*.

Use the Scale submenu to scale the image to the range of display colors. Any of a variety of linear and non-linear scaling algorithms can be applied simply by clicking on the appropriate submenu button. When scaling is performed, the range of image values currently visible within the area of the display window will be assumed. Limits on the range of image values and other scaling parameters may be input using command line switches. For long integer and floating point data, *SAOimage* uses two stages of scaling, as described below.

3.1 Range of image data versus size of color map

Image data may have a nearly infinite range of data values. Data values may be very large or very small, they may be positive or negative, and they may be real or integer.

A typical workstation can only display 256 colors on its screen at any one time. Some of the 256 colors are needed for coloring such things as the mouse cursor, text characters, background areas, and window borders. On a 256 color workstation, *SAOimage* typically uses 200 colors to display images.

For purposes of display, each pixel in the display image is assigned one of 200 integer values in the range of 0 to 255. Each of these 256 possible values is the index of an entry in the display's color map table. The color map table associates specific colors with the 256 display values (e.g. 0=black, 1=dark blue, etc.). (See section 5 for a detailed discussion of color map manipulation.)

3.2 Basic scaling theory

Take the case of an image with a range of values from 0 to +1199 (a range of 1200), and an image display with 200 colors. *SAOimage* might assign one color map entry to each range of six in the image. Thus values 0 to 5 would be assigned the lowest color map entry, 0. Values 6 to 11 would be assigned to 1, and so on up to the values between 1195 and 1199 going to entry 199 in the color map. Then, if entry 0 was black, all pixels in the image with data values between 0 and 5 would appear black in the display. If color map entry 131 was assigned the color pink, then image pixels whose data value was between 787 and 792 would appear pink in the display.

3.3 Scaling distributions

The above example is an example of linear scaling. But suppose that it was important to be able to see the differences among levels 2, 3, 4, and 5, and not as important to see a difference between 1001 and 1002. Perhaps it would be sufficient to see differences between 1001 and 1100, while 1001, 1002, 1003, etc. could all have the same color. Then it would be better to use a scaling that used more color map entries for the lower image values than the higher image values. Log and square root scaling do that.

To understand how that works, imagine a graph with image values along the side and color map values along the bottom. A plot of linear scaling would be a straight line with a constant slope. In other words, each n consecutive image value units would map to m color map levels. A log line would be curved, having less slope at the bottom (more color map values per image value range) and get steeper toward the top (fewer color map values per image value range). The square root function would be similar but follow a different curve.

3.4 Non-linear distributions

Sometimes weighting toward one end or the other isn't good enough. Perhaps the data values fall in clusters with big gaps in between. Imagine an image with most of its data clustered between 0 and 200, but two or three pixels with a value of 1000. In the linear scaling 80% of the color cells would be used for non-existent values between 200 and 1000. Worse, as is common with CCD images, there may be bad pixels with values like -1200, which have nothing to do with image source values.

3.5 Histogram equalization

A histogram is a plot with the range of possible values (divided into discrete bins) on one axis and the frequency of the actual occurrence of each value, or value within the range of each bin, on the other axis. The image histogram could be divided into as many bins as colors in the display, with each bin representing the range of image values associated with a single color in the display. The frequency value of each bin would be the number of pixels in the display that used that color.

In a poor scaling distribution, many of the bins (ranges of image values) have few or no actual occurrences in the image.

Histogram equalization is a process of adjusting the ranges of the bins such that each bin has about the same number of image pixels. If the image has many pixels with values between 100 and 200, then histogram equalization would allocate many small bins to that range, (e.g. 100–110, 111–120, etc.). If the image has relatively few pixels with values between 300 and 400, histogram equalization would allocate few large bins to that range (e.g. 300–350, 351–400). The object of histogram equalization is to maximize the information in the display by optimizing the usage of the available colors. It usually produces a dramatic improvement in the amount of visible detail in the displayed image.

The drawback of histogram equalization is that the ranges can vary greatly in size. A difference between two adjacent color map values may represent a big or a small difference in data values, with much irregularity from one step to the next.

3.6 Windowing the scaling

Often, most of the data falls within a single contiguous range, with bad pixels or infrequent events in the extremes to either end. It may be useful to restrict the scaling algorithm to a specific range, with all values below that range having the same color as the minimum and all values above that range having the same color as the maximum. This can either be accomplished by specifying the scaling limits directly, or by looking for a section of the image which is free from extreme values, and basing the scaling on the values in that section only.

3.7 Wrapping the color map

Another trick to show more detail with a limited color map is to reuse it. By wrapping the color map on itself, it could represent the range of values from 0 to 99, and then start over, 100 having the same color as 0 and 199 having the same color as 99. This works for smoothly varying data and even has a sort of contoured look to it. Where the data fluctuations are greater than the range covered by one wrap of the color map, the result is a lot of meaningless noise.

3.8 Scaling implementation and efficiency within SAOimage

In order to realize reasonable execution times for rescaling the image display, *SAOimage* does not apply its scaling functions to real data. Where the image data was real, double, or long, the data is linearly rescaled to integer values between -32767 and 32767 (a range of 65,535). Each scaling function produces a 65,535 entry lookup table for this range of values which is used to draw or redraw the display. If the data has such great extremes that this range linearly applied to the data's range will be insufficient, windowing limits for the initial linear scaling can be given on the command line using `-rmin` and `-rmax`.

The scaling function is further windowed by being applied only to the range of image values actually being displayed at the time the function is invoked. Pixels not appearing in the display window are not considered in the scaling algorithm. Thus scaling after panning and zooming will produce different scale lookup table mappings, depending on the contents of the display. Panning and zooming can be used in this way to find a better scaling.

To compensate for extreme data that may appear at the edges of an image (common in CCD images) data within 2 pixels of the edge of the display are not considered in the assessment of range. In `-verbose` mode, the scaling routine reports the range of values which it finds in the display.

One or both limits of the value range for menu-commanded scaling can be restricted by using the `-min` and `-max` arguments on the command line. When `-min` is given, the low end of the range will not extend below the `-min` value. When `-max` is given, the upper end will not extend above the `-max` value. Limit values are given in terms of original file values. The user need not know about the internally used short integer values. The `-min` and/or `-max` value can be cleared by giving the `-min` or `-max` switch with no argument.

Panning or zooming after a scale map has been made does not cause a new scale map to be calculated. The display is drawn with the existing scaling. If the new display has values outside the range when the map was made, these values are usually clipped (mapped to the color map minimum or maximum). Wrapped scaling is an exception, in that the color map wrapping is applied all the way to the maximum possible value.

3.8.1 Linear scaling

Linear scaling is fairly simple. The range of image values is divided by the number of color map values to determine the range of data values for each color map value. The fixed range is applied to mapping all image values between the minimum and maximum image value. Values below the minimum are all mapped to the lowest color map value. Values above the maximum are all mapped to the highest color map value.

3.8.2 Wrapped linear scaling

Wrapped linear scaling, as described above, divides the range of data values by the number of times the color table will roll over (the default is 10) and the result is divided by the number of color map values to determine the range of data values for each color map value. Values below the minimum are all mapped to the lowest color map value. The mapping continues to be rolled up to the highest map value (= 32767; see `-wrap` in section 1).

3.8.3 Log scaling

The distribution of data values to color map values is based on the distribution of e^n from 0 to X, where n is a parameter (the default is 10.0) and X is determined by n and the two ranges (data and color map). Positive values of n favor the lower data values, while negative values of n favor the higher data values. Values below the minimum are all mapped to the lowest color map value. Values above the maximum are all mapped to the highest color map value (see `-log` in section 1).

3.8.4 Square root scaling

The distribution of data values to color map values is based on the distribution of $X^{1/n}$ from 0 to 1, where n is a parameter (the default is 2.0) and X varies from 0 to 1 in steps determined by the number of color map values. Values of n greater than 1 favor the lower data values, while values of n less than 1 favor the higher data values (negative values and 0 are not allowed). Values below the minimum are all mapped to the lowest color map value. Values above the maximum are all mapped to the highest color map value (see `-sqrt` in section 1).

3.8.5 Histogram equalization

The principle of histogram equalization is described above. The histogram equalization algorithm in *SAOimage* differs from common one-pass algorithms by accounting for disproportionately large occurrences of one or a few image values. In an image with 1000 pixels, 500 of which have the value 32, a one-pass algorithm, given 100 colors, will try to get 10 pixels for each color. It will end up with 49 unused colors, since one color covered 500 pixels. *SAOimage*'s algorithm detects the peak (or peaks) in the histogram and allocates the remaining 99 colors among the 500 remaining pixels (5 pixels per color).

3.8.6 Image scaling by IRAF

Images sent directly by IRAF are already scaled to the range 1-200. IRAF's own scaling defaults to a windowed linear scaling where the window is determined by fitting a straight line to a small, 200 pixel, subsampled histogram. Further rescaling within *SAOimage* is not very useful (see the IRAF section).

4 Blinking between different stored display images

The display, as it currently appears in the main display window, can be stored by your workstation's display server by clicking on the blink button in the Scale submenu. Three separate images can be saved, each one associated with a different mouse button. Each saved display is associated with the same mouse button used to click on the blink button, and replaces any previous image associated with that mouse button. The saved display image can be redisplayed when in Scale mode, by pressing its mouse button while the mouse pointer is in the display window. *(This mechanism is admittedly not intuitive. Suggestions from any human-factors types are welcome.)*

Because the blink image is saved at the display server end of the X window system, it can be redisplayed much more quickly than the normal image drawing used by *SAOimage*. You will see the difference when you release the blink button. To blink between saved images, press down on another mouse button, before releasing the earlier button.

The blinked image uses the existing color mapping. The colors are not restored to any previous setting. Therefore, one should keep the same color map settings when choosing displays for blinking, or manipulate the color map to find a setting which works for all of the blinking images.

Saved display images may all be of the current image, or, more likely, may be made from different, successively loaded images. The saved image can only be used for blinking. Displaying a saved image does not re-enable any other *SAOimage* functioning for that image. To be able to use the other functions on an image which is not the most recently read image, the image must be reloaded from its file (or IRAF).

All saved displays must match the current display window dimensions. *SAOimage* unsaves all saved displays when the display window is resized.

Not all X servers are guaranteed to support display saving (called *pixmaps*). *SAOimage* will attempt to fall back on using image buffers within *SAOimage* for saving image displays. Problems may occur when the user tries to mix halftone and color pixmaps. (*Some early X server implementations were known to perform the pixmap storage one pixel at a time in 20+ minutes or simply crash. Hopefully things have matured since those days.*)

5 Color under the X window system on color workstations

Most color workstations use an 8-bit color map, making it possible to choose a palette of 256 colors and to map each pixel on the screen to one of the 256 possible colors. Each of the 256 colors in the palette can be any color specified by its red, green, and blue intensities. By contrast, early IBM PC's commonly offer palettes of 4 or 16 colors selected from a restricted group of colors. At the other extreme, 24 bit displays allow each pixel to be mapped directly to virtually any color, defined by its red, green, and blue intensities (using 8 bits for each).

X11 uses a color reservation system by which each application reserves a portion of the palette for its own use. Color reservation prevents an application from changing the colors of other applications present on the screen. Typically, *SAOimage* will be able to reserve as many as 240 color entries in the palette. However, if you (or your window manager) have tailored your X environment by specifying a variety of unusual colors for terminal windows, the clock, and other commonly used applications, there will be fewer unreserved colors in the palette. Conversely, if you are running an *SAOimage* which has reserved all of the available colors, you may be unable to bring up a new application which expects to be able to grab new colors of its own. A color in the palette is often referred to as a color cell (referring to its reservation) or color level (referring to its index between 0 and 255).

SAOimage uses color cells for more than just rendering the image. The adjustable cursor and the region cursors must also be referenced to colors in the color palette. When the cursor is drawn, pixels in the display are set to the value of the cursor color, overwriting the original image data. If no special care was taken, the image would gradually be erased as the cursor was moved about the display. To avoid this problem, *SAOimage* uses either of two strategies for tracking the cursor while it is in motion.

With one strategy, while the cursor is moving, or being adjusted, the bits in the displayed image are simply reversed to represent the cursor, and flipped back when the cursor moves away. This doesn't draw the cursor in its correct color, but, in most cases, it produces a visible cursor. When the cursor adjustment ends (the mouse button is released), the entire image is redrawn with the cursor in its new position. This same strategy is used to track the cursor on halftone displays.

The other strategy actually reserves half of the colors just for drawing the cursor. In this alternative strategy, each image color level has a corresponding level in the palette which has the cursor color. The image color levels differ from those of the cursor by one bit, which can be set and unset to represent either the cursor or the image. Drawing the cursor simply involves manipulating the distinguishing bit in display memory. In this mode, cursor can be tracked smoothly in its correct color without erasing the image data. This system is analogous to reserving *planes* of display memory to be treated as overlay planes and image planes (X does not as yet recognize true hardware overlay planes). The advantage of the plane reservation strategy is that cursor tracking is visually flawless. The disadvantage is that fewer than half of the color levels are available for rendering the image. Fortunately, it is easy to switch between the two modes (using the ovlay button), so one can choose whichever coloring strategy is most appropriate for the current activity.

5.1 Color mapping

In order to maximize the use of the available colors, *SAOimage* offers several facilities for assigning and altering colors in the palette. The association of a palette level with an image data value is handled by scaling and is explained in a separate section. There are basically three ways of assigning colors in the palette: *true color*, *gray-scale*, and *pseudocolor*. *SAOimage* supports the last. *SAOimage* also supports half-toning on non-color workstations and by selecting the mono button in the color submenu (see section 6.9).

In true color, each image data value has associated with it an actual color. True color mapping tries to associate colors, as near as possible to the true color, with each pixel in the image display. This is difficult where there are few colors in the palette. There is no support in *SAOimage* for true-color mapping.

In gray-scale, all pixels have the same color, but differ in intensity. Basically, the colors range from black to white, with shades of gray in between. It could also be done with some other color such as shades of red. The lowest data values appear black while the highest appear white (or visa-versa). The image appears as a black-and-white photograph might render it. *SAOimage* simulates this on *pseudo-color* displays but does not support X terminals with actual gray-scale displays.

In pseudo-color, any color can be assigned to any level, but all pixels with the same value will have the same color. Typically, one might use an analogy with heat, mapping the low values as shades of blue, the middle values in shades of red and the highest values as yellow or white. The idea is to use the colors to highlight differences among the data values. Depending on the levels which best distinguish the detail you wish to study, the shifts from blue to red and red to yellow can be placed at higher or lower image data values and closer together or farther apart. The changes in color can be made gradual or sharp.

Color maps may simply be a list of colors for each level, or may be created by specifying a few colors and levels and interpolating to assign colors for the in-between levels. *SAOimage* uses the latter. If one were to graph the color map, having intensity of color on one axis and palette level on the other, the graph would have fixed points with ramps or steps between them. The simplest gray-scale has no intensity for any color at one end of the palette and full intensity for all colors at the other end of the palette, with a straight line representing the interpolated colors in between. The color graph is in fact physically drawn by *SAOimage* in a separate window, and can be directly manipulated.

SAOimage has a basic gray-scale and several pseudo-color maps available in the cmap page of the Color submenu. Once you have selected a color map, you may choose to manipulate it, as described below. Reselecting the same color map, or selecting a new color map from cmap submenu, sets the selected color map, eliminating any adjustments you may have made.

5.2 Color manipulation

The color graph window is normally not displayed. It is summoned (and hidden again) by clicking the mouse on the color bar next to the display window. In the graph, each color is graphed separately, with little squares to represent the fixed control points in the graph. Where two or more color lines overlap, the line (or box) appears black. One may create a new fixed point by positioning the mouse icon in the graph and pressing a mouse button. The three mouse buttons control red, green, and blue, respectively from left to right. By holding the mouse button (or buttons) down, the fixed point can be dragged anywhere on the graph. An existing fixed point can be grabbed for dragging by positioning the mouse icon over it when pressing the button.

The graph can be adjusted, en mass, by moving the mouse in the main display window with a mouse button depressed. There are two different kinds of adjustment, *threshold/saturation* and *contrast/bias*, and an intensity adjustment called *gamma*.

With threshold/saturation, moving the mouse horizontally moves the lower (*threshold*) end of the graph up or down, while moving the mouse vertically moves the upper (*saturation*) end of the graph up and down relative to the palette. With contrast/bias, moving the mouse along the axis of the color bar shifts the entire graph up or down (*bias*) relative to the palette, while moving the mouse perpendicular to the color bar moves the ends of the graph closer together or farther apart about a middle position (*contrast*). Threshold/saturation is easy to implement and common in older pseudo-color display systems. Contrast/bias corresponds more closely to the kinds of adjustments familiar to photographers. In both cases, the middle of the display window is the default graph position relative to the palette.

5.3 Gamma correction

The intensities of the colors are normally given relative to voltage applied to the color guns in the monitor. Half intensity is half of full voltage. Unfortunately, this does not really correspond to the sensitivity of the eye. Double the voltage does not seem like doubling the intensity. Half voltage on a gray-scale does not seem like a middle gray. The gray-scale seems to favor the darker shades. The relationship between voltage and perception is generally thought to be an exponential one and is represented by the symbol γ (*small gamma*).

Changing the gamma produces a non-linear (exponential) adjustment in contrast. A gamma of between 2 and 2.2 is considered correct for a typical monitor. You can play with the gamma adjustment by selecting the gamma mode in the Color submenu. Moving the mouse horizontally in the main display window with a mouse button down adjusts the gamma. The gamma values for each color are printed beside the color graph. Gamma of 1 (linear) is in the middle of the main window. The intensity adjustment is applied directly to the palette colors and does not affect the points used to map the colors. Gamma values below 1 may be useful for sharpening the contrast before making a hard copy (see section 11). You can drag beyond the main window for gamma values outside the normal range.

Normally, all adjustments are applied equally to each of the three colors. However, the adjustments can be applied to any one or two of the colors by holding down the `Control` key. Then the three mouse buttons control red, green, and blue, respectively, as in the graph window.

The invert button in the Color submenu inverts the intensities (minimum intensity becomes maximum intensity) without changing the graph points.

5.4 Updating the graph and gamma display

Drawing the graph and printing the gamma values takes up computing time and may slow the response to your movement of the mouse. Therefore, while the colors are always continuously updated, updating of the graph occurs only when you finish (release the mouse button), unless you specifically request it. The track button in the etc submenu, controls whether the color graph is updated continuously or only upon completion of the manipulation. Tracking may be temporarily activated (or deactivated) by pressing a `Shift` key on the keyboard or toggling the `CapsLock` key.

5.5 Saving color map entries and reading them back in

The current colormap can be written to a disk file, and a previously saved colormap can be read from a disk file. The format of the disk file is ASCII and can be edited if the format is followed. The file can have comments on any line, starting with a `#` symbol. The first non-comment word in the file must be `PSEUDOCOLOR`. Each color's table is defined separately. Each color's table begins with the color name `RED`, `GREEN`, or `BLUE`. The color name may optionally be followed by the word `GAMMA`, then followed by a gamma value for that color. The vertex points in the table are defined by pairs: `(level, intensity)`. The intensities range from 0.0 (minimum) to 1.0 (maximum). The levels range from 0.0 (lowest level) to 1.0 (highest level). The points must be in ascending order by level. All three colors must be described, and each color must have at least 2 points.

When a color map is being manipulated, the effective levels of points may be shifted or stretched above 1.0 or below 0.0. These points are preserved in the disk file, and when read in, they may be shifted back into the visible range. The novice user should remember that if one starts with the **A** colormap, shifts it, and then writes it out, the stored colormap is the shifted map, not the original **A** map.

6 Graphics cursors and cursor mode in SAOimage

SAOimage provides a variety of "software" cursors to identify or delineate areas of the image. These are not to be confused with the workstation's mouse cursor (the icon which moves as you move the physical mouse on the mouse-pad or desk top). The software cursors are selected and manipulated while in Cursor mode, which is activated by selecting the Cursor button in the main button menu. In this section, software cursors will be referred to as *cursors* and the mouse cursor (also called the pointer) will be called *the mouse*. Special cursor interactions which work in conjunction with IRAF are discussed in the section on IRAF.

There are seven basic cursor shapes: *point*, *polygon*, rectangular *box*, *circle*, *ellipse*, *arrow*, and *text*. The cursor type is selectable by the corresponding button in the cursor button submenu (see section 2). The box, circle, and ellipse cursor types can be used to make annuli of concentric equivalent cursors.

Only one active cursor can exist at any given time. However, cursors can be stored, written to disk, and recalled using the region features (see section 7). The cursor is always displayed, even when it is not active (not in Cursor mode). Since the active point cursor has no visible component, select it when you do not wish to have a cursor visible in your display.

When a cursor is selected, its position, size, and orientation are manipulated by using the mouse (see section 9). As a general rule, the left mouse button controls the position of the cursor, the middle mouse button controls the shape or size of the cursor, and the right mouse button controls the angular orientation of the cursor (if it has one). You may either *click* or *drag* the mouse. Positions, dimensions, and angles (where applicable) are shared among the latter three cursors. The point and polygon cursors pose special cases to the rule; read about them separately, below. The text and arrow cursors are described separately, at the end.

When *SAOimage* is in its `-verbose` mode (see section 1), the cursor's coordinates and dimensions are printed to the parent process's terminal window, each time a cursor manipulation is completed.

6.1 Positioning

To position the cursor by clicking, move the mouse to the pixel where you want to center the cursor and then click the left mouse button. To position the cursor by dragging, press and continue holding down the left mouse button while moving the mouse. When the cursor is positioned where you want it to be, release the left mouse button. Note that you can use the magnifier tracking feature to aid in positioning the cursor and you can use the keyboard arrow keys to aid in fine positioning the mouse (see sections 10 and 9).

6.2 Sizing

To adjust the size or shape of the cursor, the same clicking or dragging procedure is used with the middle mouse button to position the cursor's edge while its center stays fixed. When sizing the box or ellipse, certain restrictions apply, depending on the position of the mouse when you first press the middle mouse button. If you are near to a line through the center and one corner of the box, you will be controlling the location of the corners of the box (adjusting both dimensions). If you are not near one of the diagonals, you will control only the width or height, depending on the side to which the mouse is closest. The ellipse is adjusted by controlling an imaginary box which encloses the ellipse. When adjusting the size of a cursor annulus, since the ratio of width to height is fixed, the mouse controls the actual edge of the cursor, regardless of its type.

6.3 Rotation angle

The right mouse button is used to control the rotation angle of boxes and ellipses. When the right mouse button is pressed or held down, the angle is determined by a line from the center of the cursor to the mouse. The initial angle (0 degrees) points toward the top of the screen. To reset the angle to 0 degrees, click on the ortho button (0 degrees) in the Cursor submenu. For the circle cursor, the right button has no function, while for points and polygons it performs a special delete function (see below).

6.4 Point cursor

The point cursor is used to flag particular image pixels or coordinate points. Its position is defined as that of the mouse. No active cursor is drawn. The mouse buttons are mapped directly to region functions (see section 7). The left mouse button stores a point *include* region (to flag a pixel or coordinate of interest). The middle mouse button stores a point *exclude* region (to flag a bad pixel). With the mouse positioned to point at the first character in the label of a saved point, pressing the right mouse button deletes that saved point.

Saved points are represented by a label giving either its pixel coordinates or an index number. By default, the label is stenciled over the image. However, if the label is too hard to read, the background around the label can be made solid by selecting the appropriate command line option. The point is at the left edge of the point label, and on a line with the bottom of the characters (not at the lowest edge of the background). When making hard copies, *SAOimage* always fills in the label backgrounds (see section 11).

6.5 Polygon cursor

A polygon is defined by straight lines connecting a set of vertices. For the active polygon cursor, each vertex is represented by a tiny box. To add additional vertex points, drag or click with the middle mouse button. The next vertex is always added to the side nearest the mouse pointer when the middle button is depressed. To move an existing vertex, point the mouse directly at it when depressing the middle mouse button. To delete an unwanted vertex, point the mouse at the unwanted vertex and click the right mouse button.

The entire polygon can be be moved at any time by using the left mouse button. When the left button is depressed, the polygon is moved such that the nearest vertex has the same position as the mouse pointer.

Clicking on any cursor type submenu button, including the polygon, causes the polygon to be reduced back to a single point. In other words, if you switch cursor types, you cannot return to the polygon which you had been constructing, unless you saved it as a region.

6.6 Cursor annuli

Annuli of box, circle, and ellipse cursors are available by selecting the **annuli** button (concentric box icon) from the cursor submenu. Set the cursor to the desired shape and angle before selecting the annuli feature, as these cannot be changed once the annuli feature has been selected. (The annuli selection changes the functioning of the middle mouse button and disables any angle control.)

Annuli are sized by clicking or dragging with the middle mouse button. The center, angle, and width to height ratio remain unchanged while sizing. When the middle mouse button is first depressed, three possible events may occur, enabling the user to 1) create an annulus with an arbitrary radius, 2) create annuli with fixed radius increments, or 3) change the size of an existing annulus.

1. If the mouse is between existing annuli or not too far from an existing annulus, a new annulus will be created sized by the mouse position.

2. If the mouse is well outside the existing cursor annuli, a new cursor is create which radius is initially incremented from the outer annulus by the same increment as that between the outer two annuli (or the annulus and the center if there is only one annulus). A comparable situation applies when the mouse is well inside the innermost annulus. However, once the annulus is created, if you proceed to drag the mouse with the middle mouse button down, the size of the new annulus reverts to being controlled by the mouse.
3. If the mouse is pointing at the edge of an existing annulus, that annulus will be grabbed for sizing.

To delete an unwanted annulus, point the mouse at or near the unwanted annulus and click the right mouse button. The innermost or outermost annulus can be deleted simply by clicking the right mouse button while the mouse is inside or outside all of the annuli.

All of the annuli are forgotten when any cursor submenu button is selected, or reselected. The annuli button itself toggles between on and off.

6.7 Arrow cursor

The arrow cursor consists of three line segments forming the shaft and head of the arrow. The left button positions the entire arrow (from its tip). The middle and right buttons move the tail end of the arrow shaft, while the tip stays fixed. Typically, one will place the head by an object of interest and then adjust the tail for a desired length and angle. Although the length of the tail adjusts to changes in image magnification, the size of the head has a fixed screen dimension.

6.8 Text cursor

The text cursor allows you to put text strings on the image using the cursor mechanisms. The text string is fully editable, using the same line editor as used for command input (see the list of editor commands at the end of section 10). The current editor position is marked by an under-bar cursor. As with other cursors, the text cursor can be moved with the left mouse button. The middle and right buttons have no effect. The position of the text cursor is at its lower-left corner. The size of the text font does not change with changes in image magnification.

To save the string, as a region, use the `Return` key (*include* color) or the `LineFeed` key (*exclude* color). The *include* text string is drawn with a black background, while the *exclude* text string is always drawn with a white background. For hard copy, this allows you to choose white-on-black or black-on-white (see section 11).

6.9 Cursor color

Tracking a cursor means updating the displayed cursor as its shape, size, or location is being manipulated, enabling the user to see the effect of the manipulation. Tracking a visible cursor across the display poses a special problem; how to quickly draw and undraw the cursor as it moves, without wiping out the image underneath. *SAOimage* uses two alternative mechanisms to do this.

On halftone workstations, the tracking cursor is drawn simply as the opposite color (*exclusive or*) of the image. Black pixels appear white and white pixels appear black as the cursor tracks across them. When the tracking action is completed (when the mouse button is released), the entire display is redrawn, along with the new cursor.

The same mechanism, as that used on halftone systems, is used on color workstations when a large number (more than 100) of display color levels is needed. During tracking, the inverse of the display pixels may not result in very noticeable colors, but at least the *shimmering* of the display, as the cursor moves, makes it possible to see where the cursor is.

When many cursor manipulations are to be performed, a better alternative is available. One bit in each color value is reserved for the cursor. Then the cursor may be drawn and undrawn without affecting the display. This results in a smooth, continuous display of the cursor throughout any tracking interaction, with no need to redraw the image at the completion of the action. The disadvantage is that the number of available colors for the image is cut by a factor of two (plus two more colors for the saved regions and other overlay graphics). (*Note: Some versions of Sun's OpenWindows do not correctly undraw the text cursor in this mode.*)

On color workstations, one may switch freely between the overlay and non-overlay modes by using the ovlay toggle button in the Color submenu. See section 5 for more details.

7 SAOimage cursor regions

A region is a cursor which has been stored. *SAOimage* cursors can be stored, recalled, written to disk, and read from disk using the region features. Region controls work only when *SAOimage* is in cursor mode. Regions were designed for use with SAO's PROS/IRAF image analysis software, but they can be generally useful.

There are two classes of regions; *include* and *exclude*. When used by the PROS software, an *include* region indicates an area of interest for analysis (e.g. for counts within a region). An *exclude* region indicates areas or pixels to be excluded from analysis (e.g. bad pixels or areas affected by other phenomena such as an adjacent source).

7.1 Saving and unsaving

To save the current cursor as a region, type the '`S`' key for an *include* region or the '`E`' key for an *exclude* region. To unsave the most recently saved region, type the '`Delete`' key. To unsave an arbitrary region, place the mouse within that region and type the '`D`' key. If the mouse is within two regions when the '`D`' key is typed, the smaller region is deleted. Exceptions to these rules apply to the point or text cursor types.

When a point cursor is the active cursor, the three mouse keys are mapped to perform the same functions as the '`S`' key (left mouse button), the '`E`' key (middle mouse button) and the '`D`' key (right mouse button). Only point regions can be deleted with the right mouse button. For purposes of deleting, the area of a point region is defined to be the area of the first character of its label.

When a text cursor is the active cursor, the '`S`', '`E`', and '`Delete`' keys have the above functions only when the mouse is outside of the main display window. With the mouse inside the main display window, all keys are used for text entry and editing. Pressing the '`Return`' key saves the text string as an *include* region, while pressing the '`LineFeed`' key saves the text string as an *exclude* region.

In either of these cases, the cursor position is advanced for starting a new line of text. See below for special coloring differences which also apply to text regions.

Saved regions are normally drawn and labeled. On color workstations, *include* regions are drawn in yellow, while *exclude* regions are drawn in red. The label usually has three parts: the coordinates of the center, a line from the center to the edge along the region's positive Y axis, and the length of the radius along that line. The radius is followed by a '`+`' for *include* regions or a '`-`' to indicate an *exlude* region. Only the center coordinate is given for a point (followed by a '`+`' or '`-`'). The point is at the left edge of the label and on a line along the bottom of the label characters. Polygons have no labels.

The text region also has a background. For the *include*, this background is black, while for the *exclude*, the background is white. When making a hardcopy, the *include* text region is drawn white on black, while the *exclude* text region is drawn black on white.

7.2 Menu controls

In the Cursor submenu, the region button toggles between the cursor control page and the region submenu page.

In the region submenu page, the label button suppresses labeling of the regions. The view button completely suppresses display of the regions. These buttons are both toggle buttons which are turned on or off when pressed.

The cycle button makes the cursor an exact copy of a region (*recalling* the cursor). The center of that region and its sequence number are displayed in the magnifier window. Repeatedly pressing cycle, copies each region in turn (*cycles through* the regions). The cycled cursor is a normal cursor and can be manipulated. The copied region is unaffected by cycle. However, the just copied region can be deleted by pressing the omit button. The omit button only works if the cursor has just been cycled to that region. Once you alter the cursor, omit is disabled. To adjust a region, cycle the cursor to that region, omit the original region, adjust the cursor, and then save the cursor as a new region.

The reset button clears the memory of all regions (*deletes* all regions).

The read and write buttons read and write ASCII descriptions of regions to or from a disk file.

7.3 Regions as ASCII disk files

After saving one or more regions, the stored regions can be written to a disk file. Normally the regions are written in a format which can be parsed by the PROS/IRAF software. If *SAOimage* is in `-imtool` mode, it mimics IRAF's *imtool* output, which is simply a list of coordinates for the centers of cursors. Either type of file can be read at any time, regardless of the mode.

A PROS region consists of a leading space (*include*) or '-' sign (*exclude*) followed by the name of the region type, its center coordinates in file pixels, and then its radius on its X axis, its radius on its Y axis, and its angle in degrees (counter clockwise from straight up as per astronomical convention). Some regions don't have angles (or the angle 0 is not given), circles have only 1 radius, and points have no radii. Polygons are represented by pairs of coordinates (X,Y,X,Y,...). Arrows and text are also saved, but appear as comments. While *SAOimage* can read these back in, they will not confuse other programs which are interested only in the spatial regions. See the PROS region documentation for a full explanation.

```
# images/m51.fits
# Mon Jan 16 16:09:43 1989
# shape x, y, [x dimension, y dimension], [angle]
 BOX(135.67,213.00,18.00,10.67)
 BOX(504.00,538.00,16.96,106.53,290.714)
 CIRCLE(129.00,306.67,9.89)
-ELLIPSE(486.00,468.00,36.53,44.08,329.036)
-POINT(83.00,349.00)
 POLYGON(131.00,245.00,120.67,280.33,165.00,245.00,140.00,234.67)
```

Annuli of boxes and ellipses are not supported basic PROS region types. They are implemented as logical combinations of the basic PROS types.

```
 BOX(149.67,205.00,10.00,4.00)
 BOX(149.67,205.00,20.00,8.00) & !BOX(149.67,205.00,10.00,4.00)
 BOX(149.67,205.00,30.00,12.00) & !BOX(149.67,205.00,20.00,8.00)
```

The above example defines a center area surrounded by two concentric rings.

When write is pressed, a popup window appears to prompt the user for the filename. The window may be initialized to a default, or the previously given filename. To select a different name, edit the name in the window (see section 10). If the named file already exists, another popup window appears with the options to overwrite the existing file (`o`), append to the existing file (`a`), or quit without writing anything (`q`).

PROS format files which were previously written can be read back in. Press the read button and give the name of the file. Reading does not support the full PROS region syntax (i.e. PIE slices are not supported), but much of the syntax is supported, including any syntax used by the write. Regions read from disk are added to any regions already stored.

The output as an *imtool* list, equivalent to the first example, would look like the following:

```
# images/m51.fits
# Fri Jan 20 13:35:36 1989
# (x, y)
136 213
504 538
129 307
486 468
83 349
131 245
149 205
```

If your environment has been properly initialized as for an IRAF *imtool* user, the default list output filename will follow standard IRAF conventions. When read by *SAOimage*, the list coordinates are all read as *exclude* points.

8 Image panning and zooming in SAOimage

The loaded image may be panned and/or zoomed for its display in the central display window. Panning and zooming are interactively controlled by clicking or dragging the mouse in the *pan window*. The pan window is the small image window at the top of the desktop, next to the magnifier.

The pan window shows the entire area of the loaded image. (By default the image in the pan window was reduced by taking the maximum value in each block of the original image, but averaging, summing, or subsampling may be used instead—see section 1). A box cursor in the pan window shows the area of the image currently being displayed in the main display window. This box can be moved and sized just like the regular box cursor (but without angular rotation control).

The left mouse button controls the location of the center of the box (the display image). The middle button controls the edge of the box, for the given center, thus determining its size. The size of the box is restricted to integer zoom factors. (The actual algorithm choses the smallest zoom that still includes the mouse pointer's position.) With a three button mouse, one can switch from one type of control to the other, by holding the first button down, until after the second button has been pressed. The interaction can be cancelled by dragging the pointer outside of the pan window's borders. Only after the last button is released is the main display redrawn, with the indicated pan and zoom.

When the mouse is in the pan window, the area under the mouse pointer can be magnified in the magnifier by pressing or holding the shift key on the keyboard. The `T` key table also works with the pointer in the pan window (see section 10). As anywhere else, the arrow keys can be used for fine positioning.

In pan mode, the same types of interactions as those used in the pan window can be performed in the main display window. Clicking or dragging with the left button controls the center of the display. Clicking or dragging with the middle button determines the zoom factor. The box cursor in the pan window tracks these manipulations as the mouse is dragged in the display window, just as it does when it is being directly manipulated in the pan window.

The pan submenu allows a more basic method of changing the zoom (e.g. x4 or x1/2). The center button centers the display on the image and the zoom 1 sets it to one display pixel per image pixel.

9 Using the mouse in SAOimage

As with most interactive workstation programs, *SAOimage* uses the mouse and its buttons for many of its interactions. To take full advantage of *SAOimage*'s features, the user must understand how to use the mouse. There are three basic types of mouse interactions: moving, clicking, and dragging. Moving means moving the mouse while keeping it flat on its mouse pad or desk surface. Clicking means pressing one of the buttons on the mouse and then releasing it. Dragging means holding one or more of the mouse's buttons down while moving the mouse. Rotating the mouse is not a meaningful action.

The mouse is represented on the screen by an *icon*, a simple symbol about a half centimeter on a side, which moves on the screen as you move the mouse on your table. The box with buttons which you hold in your hand and the icon on the screen are interchangeably (or in concert) referred to as the mouse. In X11 literature, the mouse is commonly referred to as the *pointer*. The mouse *occupies* a window or object when its icon on the screen is within the borders of that window or object.

Within *SAOimage*, the mouse interaction (as well as the keyboard interaction) is generally governed by the mode of the main menu (see section 2) and the window or object which the mouse occupies. This means that the response to actions taken with the mouse in one window may differ from the response when the mouse is in a different window. The window associated with a particular mouse interaction is called its *focus*. In some cases, it may be possible to continue a *dragging* action beyond the edge of a window without shifting the focus to another window.

Some window managers require the user to click on an application's window before that application can receive any events. This is called *click-to-focus*. Commonly, this initial click is also passed to the application as an event. Since you may not have intended anything more than to set the focus, remember to do your *click-to-focus* in the inactive, upper-left, region of *SAOimage*'s main window, or better yet, change the window manager's *click-to-focus* behavior to *focus-follows-mouse*.

The mouse icon has a point (called the *hot spot*) which is used to determine its screen coordinates. This point is usually in the center of the icon or at a corner. The shape of the icon should make it easy to guess where the hot spot is. The mouse location (or *where the mouse is pointing*) refers to the hot spot. Often dragging is initiated by pointing the mouse at a particular object (i.e. the vertex of a graph or polygon) as the appropriate mouse button is pressed, and then dragging that object while continuing to hold the mouse button down.

For machines with a two-button mouse (e.g. IBM PS/2) the middle button is indicated by holding both buttons down at the same time. On those machines, the initial response to a mouse button has a noticeable delay to smooth over differences in depression time of the two buttons, should a two-button event be intended. On machines with a three-button mouse, some interactions may be performed while dragging with more than one mouse button down at the same time. In the color graph, for example, where each mouse button corresponds to one of the three display colors (red, green, blue), one can manipulate all three color simultaneously by dragging a vertex while holding all three buttons down.

Some actions perform a complete response only after the last mouse button is released. For example, in the pan window, you may manipulate both the magnification and the center of the main display window (zoom and pan). The main display window is not actually redrawn until all mouse buttons are released. The area that would appear in the main display window, if the mouse buttons were released at that point, is represented by a box drawn in the pan window. With a three-button mouse, you may switch directly from adjusting the zoom to adjusting the pan (or visa-versa) by pressing the other mouse button down while still holding the former button down.

9.1 Tracking

Some interactions actually *track* the mouse. By this we mean that which is being controlled is repeatedly updated as the mouse is moved, giving the impression of smooth, continuous change. An example of this is moving a cursor with the mouse. As you move the mouse, the displayed cursor moves across the screen. Another example is the magnifier window. As you move the mouse, the view in the magnifier shifts (or pans), as if watching moving scenery.

While tracking is generally desirable, your workstation processor may not be fast enough to make many repeated updates for the illusion of smooth continuous change. This is especially a problem for complicated processes (e.g. redrawing the main display window or drawing several elliptical annuli) or when many things are all tracking at the same time (e.g. changing the colors while also updating the graph which represents the color map). When this happens, the effect of moving the mouse may lag annoyingly behind the actual movement of the mouse.

Recognizing that some processors are slower than others, *SAOimage* allows some tracking actions to be enabled or disabled. The magnifier window can track the mouse as it moves across the main display or pan windows. The color table graph can track changes to the color map as they are controlled either in the main display window or directly on the graph (see section 5). Both of these tracking functions are enabled or disabled by the `track` button in the `etc` button submenu. The `track` selection can be overridden (reversed) temporarily by pressing the `Shift` key or more long term by toggling the `CapsLock` key. The running text display of the mouse coordinates and the corresponding pixel value can be enabled or disabled by the `coord` button in the `etc` button submenu. This is also overridden by the `Shift` and `CapsLock` keys.

9.2 Fine movement

SAOimage uses the user's default mouse movement velocity settings. To move the mouse one pixel at a time in any direction, use the keyboard *arrow* keys (see section 10). The arrow keys can be used at any time in place of physically moving the mouse. Thus to perform a fine movement while dragging, hold the mouse steady with the mouse button (or buttons) depressed with one hand, while pressing the appropriate arrow keys with the other hand. If you are not already tracking, you may wish to hold the `Shift` key down as you do so, to better see the effect of the fine movements.

10 SAOimage keyboard input

Many of the keys on the keyboard are mapped to perform useful functions. Striking the key results in an immediate response. The response may be dependent on the window in which the mouse cursor is positioned. Some responses are also dependent on the main menu mode (e.g. `Cursor`). Both upper and lower case are mapped the same.

In addition to the instant action keys, the `Control`, `Shift`, and `Meta` (left and right) keys change the response to mouse actions when held down while performing the mouse action.

Sometimes, *SAOimage* will ask for text or numeric input with a pop-up window. Input in this situation is as one would expect and is unaffected by the mappings which are otherwise in effect. The cursor read-back mode of IRAF also overrides many keyboard functions.

10.1 Key commands

`Up Arrow`, `Down Arrow`, `Left Arrow`, `Right Arrow`: The four arrow keys can be used to move the mouse cursor one screen pixel in the given direction. The mouse interaction is exactly as if you moved the mouse by hand. This facilitates fine positioning and works in all windows and all modes.

`R:` (*raise*) Raise all *SAOimage* windows to the front of any other windows on the screen. The '`A`' key also performs this function.

`A:` (*refresh*) Raise and redraw the display windows. This is equivalent to `Ctrl-L` in a character terminal window.

`N:` (*new*) enter new command-line arguments. This brings up a popup window for text input. Use the same switches and arguments as you would on the command line (see COMMAND LINE). All of the command line arguments which affect selection and display of an image can be used. Some configuration arguments (e.g. `-display`, `-geometry`, and `-gd`) are not effective at this time.

`T:` (*table*) When the mouse is in the main display window or the pan window, this prints a table of the pixel values surrounding the current mouse position. The table is printed in the terminal window from which *SAOimage* was called. If known, the original file values are shown. Otherwise, scaled approximations are used. (`L` has a seemingly similar effect but is useful only for debugging purposes).

`S:` (*save*) When in cursor mode, and the mouse is in the display window, the current cursor is saved as a region (of type include).

`E:` (*exclude*) When in cursor mode, and the mouse is in the display window, the current cursor is saved as a region (of type exclude).

`D:` (*delete*) When in cursor mode, and the mouse is in the display window, the smallest region which encloses the mouse is deleted. In the case of a point region, its area is defined as the area of the first character of its label.

`Delete:` When in cursor mode, and the mouse is in the display window, the most recently saved region is deleted.

10.2 Modifier keys

For an explanation of mouse clicking and dragging, see section 9.

`Shift:` Holding the `Shift` key down toggles the track and coord tracking actions.

If no tracking was selected and the mouse is in either the main main display window or the pan window, touching the `Shift` key will update the magnifier window to the current mouse location. If the mouse is in the main display window, the coordinates and pixel value will also appear in the upper right, above the buttons.

If you are in Color mode and dragging the mouse with a button down in either the main or color graph window, the graph will be updated to the current state of the color table.

By contining to hold the `Shift` key down, tracking will continue to update as you move the mouse.

If you were tracking before touching the `Shift` key, pressing the `Shift` key will suspend tracking, enabling you to preserve the existing tracking displays while moving the mouse.

`CapsLock`: Toggling to caps mode by using the `CapsLock` key has the same effect as holding the `Shift` key down. When using this option, you may need to toggle it off to make normal text entry in the pop-up or a terminal window.

`Control`: When manipulating the color map in the main diplay window, holding the `Control` key down, restricts the effect to only the color(s) corresponding to the mouse button(s) being pressed (left=red, middle=green, right=blue).

`Meta`: The `Meta` keys have traditionally been used for window manager functioning. *SAOimage* normally ignores keyboard and mouse input while a `Meta` key is held down.

10.3 Editable text input

When *SAOimage* needs string input, such as the name of a file or new command line arguments, it presents a pop-up window for you to type in. The line starts with a default already typed in. Many keyboard editor keys are recognized, including the arrows. The popup window also recognizes many Emacs style line edit commands. The input is taken when the user strikes the `Return` key.

`Ctrl-A`: Move text cursor to beginning of line

`Ctrl-B`: Move the text cursor back one character (backspace)

`Ctrl-C`: Abort the interaction and return to *SAOimage*

`Ctrl-D`: Delete the character under the text cursor

`Ctrl-E`: Move the text cursor to the end of the line

`Ctrl-F`: Move forward one space

`Ctrl-G`: Clear escape if just typed

`Ctrl-K`: Delete all characters from the text cursor to the end of the line

`Ctrl-N`: Recall next input line (clear input line if no next in buffer)

`Ctrl-P`: Recall prior input line

`Delete`: Delete character before the text cursor

`Esc-B`: Move to beginning of previous word

`Esc-D`: Delete to end of next word

`Esc-F`: Move to end of next word

For numeric input, the typed input must be in acceptable integer or real (where allowed) format. Multiple entries, when desired, must be separated by spaces, commas, or tabs. (The line is read using scanf).

The above edit commands, with the exception of `Ctrl-C`, `Ctrl-N`, and `Ctrl-P`, also apply to the text cursor.

11 Outputting an image to a laser printer

Clicking on the print button in the etc submenu dumps a hardcopy of the main display image to a PostScript printer. The output image includes the central display window and the colorbar. Any cursors showing in the display window are included in the output. This is a monochrome file, not color PostScript.

In order to get a hardcopy that looks something like what you have on the screen, it is best to use the gray colormap. You should find a good scaling for the image and manipulate the contrast with the gamma control. The output image will be dithered by the PostScript printer's own dithering algorithm.

On color workstations, the image is read from the screen and includes any graphics which may be showing. The image also includes the upper portion of *SAOimage* (the title area and two auxiliary windows). By default, the buttons are not included, but they may be included by first using the `-lprbuttons` switch on the command line.

Cursor and region graphics are colored, white of black, to contrast with the lowest image value's color (the low end of the color bar). Saved text cursors are colored white-on-black or black-on-white depending on whether they were saved as *include* and *exclude* regions.

When *SAOimage* is in its mono (halfone display) mode, an image of the display window, only, is created and printed. This image is made directly from the internal data, using the current scaling information, and does not include any cursor graphics. (It will include markings made with IRAF 's *tvmark* task).

The output file can be directed anywhere by setting the `R_DISPOSE` environment variable. The `R_DISPOSE` string is a format statement for an sprintf which creates the UNIX command, where '`%s`' will be replaced by the output file's temporary filename. The default is '`lpr -Plw -r %s`'. The `-r` in the print statement deletes the PostScript file after being printed. One could, for example, set `R_DISPOSE` to '`mv %s /tmp/foo.ps`' to save the file for later use.

If no `R_DISPOSE` environment variable is set, *SAOimage* looks for a `PRINTER` environment variable. If one is set, the hardcopy will be sent to that printer. If no `PRINTER` variable is found, the default destination is lw.

The PostScript file is scaled to fill an 8.5x11 inch page (regardless of the printer resolution). To get a smaller image, centered on the page, see the comment included in the ASCII section at top of the output file.

Sending an image to a printer over serial lines at 9600 baud, as is common for some Apple printers, will tie up the printer for about 12 minutes. Be considerate of other users who may wish to use the printer.

While this is generally a sufficient and convenient form of hardcopy, the user may also wish to try *xwd* or one of the other window dump facilities available for X11.